SOFTWARE MUSIC SYNTHESIS SYSTEM

For 8080, Z-80, 8085
based microcomputers

by JON BOKELMAN

CALIFORNIA SOFTWARE CO.
CALIFORNIA SOFTWARE CO.
CALIFORNIA SOFTWARE CO.
CALIFORNIA SOFTWARE CO.

SOFTWARE MUSIC SYNTHESIZER SYSTEM

by Jon Bokelman

Published by
California Software Co.
P.O. Box 275
El Cerrito, CA 94530

TABLE OF CONTENTS

SOFTWARE MUSIC SYNTHESIZER SYSTEM

INTRODUCTION

This manual describes the assembly, configuration and use of the Software Music Synthesizer System (SMS). The Table of Contents will direct you to the major sub-sections within those catagories.

SMS has been designed to run on a wide variety of micro-computer systems. The three versions covered by this manual are: CP/M, SOLOS/CUTER and North Star DOS. (CP/M is a trademark of Digital Research; SOLOS and CUTER are trademarks of Processor Technology.) SMS will run with any other system insofar as it can be made to "look" like one of these operating systems or monitors.

Occasionally, the material covered in this manual does not apply to all versions of SMS. All such sections will be so indicated and you may skip over any portions that do not apply to your computer system.

CIRCUIT BOARD ASSEMBLY

Skip this section if your SMS interface board is pre-assembled.

The SMS interface circuit is very simple and assembly of the interface board is straightforward. The location of each component is clearly shown on the diagram in the appendix. Take care that the parts are positioned correctly before they are soldered in place.

Mount the 8 position DIP switch with switch #1 (most significant bit) at the bottom or left side (see diagram).

The resistors supplied with the kit may be either discreet components or in DIP packages. DIP resistors should be mounted with pin 1 (indicated by a notch or a dot) to the left.

The ribbon cable header supplied with the kit may contain fewer pins than there are holes on the pc board. (The pc board will accomadate a 40 pin header.) Always mount the header with the left-most pin (pin 1) in the left-most hole on the pc board.

DO NOT mount any components in the three rows of holes reserved for jumpers (see diagram in Appendix).

INTERFACE STRAPPING

Skip this section if your SMS interface board is pre-assembled and pre-strapped for your particular computer system.

Interfacing a peripheral to a micro-processor system is difficult. There is no standard connector, although D-25 is common, and there are no standard pin assignments. Strapping involves matching the bits of your computers I/O ports to corresponding bits on the SMS interface board.

In addition, for your assembled interface board you will need a plug(s) that mates with the parallel I/O connector(s) on your computer, some ribbon cable, a connector to mate with the header on the board and some insulated wire (24 gauge) for jumpers. All these items can be obtained from electronic parts suppliers or computer stores. Many suppliers will custom make cable assemblies according to your instructions. The ribbon cable should be kept as short as possible (2-4 feet.)

You will also need the pin assignments for your computer's parallel I/O ports, an ohm meter or a continuity test light will be very helpful and some patience. The task is not difficult, it's just confusing because the different kinds of connectors use different numbering schemes and pin #13 at one end of the cable may turn into pin #2 at the other end.

On the interface board locate the two rows of holes labeled 1 to 40. These holes correspond to the conductors of the ribbon cable with #1 being the left-most conductor, 40 being the last.

Next locate the holes marked I0 to I7 (see diagram) , GND and O0 to O7. I0 is to be jumpered to the ribbon cable conductor that ultimately connects to bit 0 of your computer's INPUT port Similarly, I1 to I7 correspond to INPUT port bits 1 to 7. O0 to O7 should be jumpered to correspond to bits 0 to 7 of your computer's OUTPUT port. In all cases, bit 0 is the least significant bit (LSB) and bit 7 is the most significant bit (MSB.) (There is no standard bit numbering convention, either.) GND should be jumpered to any conductor that connects to a signal ground at the computer end.

The SMS interface board does not require any data ready strobe or other hand-shake signals.

Now go back and check everything again. Trace each signal path with an ohm meter, if possible, or have someone else check your work.

Finally, connect a shielded audio cable to the holes marked AUDIO. The center conductor is '+' and the braided shield is '-'. Using a piece of wire or a twist-tie secure the audio cable to the large hole at the edge of the board.

To prevent the sharp wire stubble on the back of the board from scratching your table top, tape or glue a piece of heavy cardboard to back of the interface board.

SYSTEM INSTALLATION

Attach the interface cable you have prepared to the interface board and to the I/O port connector(s) on your computer. (The computer should be turned off before making or breaking any connections to it.) Connect the audio cable to the AUX or TAPE input of an amplifier or receiver. DO NOT use the phono input. (Again, the amplifier should be off or at minimum volume before making or breaking any connections to it.)

Turn on the computer and the amplifier and slowly turn up the volume. If you hear a loud hum, double check all the connections and the interface jumpers. Pay particular attention to all the ground connections. If you hear anything else, double check the interface jumpers and make sure you are connected to the proper I/O ports.

If you don't hear anything, briefly touch the '+' connection on the interface board with your finger. This should produce some hum. If it doesn't, double check all the amplifier and speaker connections and make sure the correct input is selected.

SYSTEM GENERATION

The SMS software, like the interface board, must be tailored to the computer system it will run on. The I/O port addresses must be patched and various tables must be modified to accomatate different micro processor clock rates. However, unlike the hardware, all the calculations and patching are done for you by the system generation program.

The system generation program performs two major functions. First, it allows you to tailor the SMS program to your particular micro processor installation and, second, it allows you to modify the standard sine wave tables to produce different kinds of sounds.

The generation procedure consists of a programmed sequence of steps. At each step SMS will ask a question and wait for your answer. The range of acceptable answers will be displayed with each question. After typing your answer press the 'carriage return' or 'enter' key. If your answer is accepted, SMS will proceed to the next step. If not, the question will be repeated until an acceptable answer is obtained. Your answer should not contain any leading or embedded blanks.

Proceed to the System Generation section that applies to your computer system; CP/M, DOS or SOLOS/CUTER.

SYSTEM GENERATION - CP/M

These instructions assume a one drive CP/M system and that you are familiar with its use.

Boot-up CP/M in the usual way. Remove the system diskette and insert the SMS diskette in drive A. The generation program is named SMS.COM so just type SMS to load and run the program. SMS should respond with the copyright notice followed by the first question.

The first three steps involve basic information about your computer system.

Proceed to STEP 1 of the system generation instructions.

SYSTEM GENERATION - DOS

These instructions assume a one drive DOS system and that you are familiar with its use.

Bring up the DOS system in the usual way. Remove the system diskette and insert the SMS diskette in drive 1. The generation program is named SMS so just type GO SMS to load and run the program. SMS should respond with the copyright notice followed by the first question.

The first four steps involve basic information about your computer system.

STEP 0: ADDRESS LIMIT? <4CFF-E7FF>

Enter the hexadecimal address of the last byte of contiguous system RAM. SMS will not use any locations above this address. A standard DOS with system ROM starting at location E800 or E900 is assumed.

Proceed to STEP 1 of the system generation instructions.

SYSTEM GENERATION - SOLOS/CUTER

These instructions assume a one drive SOLOS/CUTER system and that you are familiar with its use.

Bring up the SOLOS/CUTER system in the usual way. Place the SMS cassette in the recorder and make sure it is fully rewound before pressing the PLAY button. The generation program is named SMS so just type XEQ SMS to load and run the program. SMS should respond with the copyright notice followed by the first question.

The first four steps involve basic information about your computer system.

STEP 0: ADDRESS LIMIT? <1FFF-FFFF>

Enter the hexadecimal address of the last byte of contiguous system RAM. SMS will not use any locations above this address. The upper address limit of FFFF is theoretical since SMS will not accept an address that is greater than the address passed to it by SOLOS/CUTER in the HL register. On a SOL, the address limit cannot be greater than BFFF.

Proceed to STEP 1 of the system generation instructions.

SYSTEM GENERATION - STEP 1

STEP 1: INTERRUPTS? <Y/N>

Certain SMS routines disable interrupts when they are running. This question asks if you want interrupts enabled at all other times. Unless you want interrupts disabled at all times you should answer Y for yes.

STEP 2: CPU CLOCK? <178-710>

Enter your CPU clock rate in hundredths of mega-hertz. For a 2 Mhz 8080 enter 200, for a 2.6 Mhz Z-80 enter 260, etc.

Some 4 Mhz Z-80 systems insert a wait state during each M1 cycle. If you have such a system, multiply the CPU clock rate by 0.908. If your system adds a wait state to every machine cycle, multiply the CPU clock rate by 0.789.

STEP 3: NUMBER OF VOICES? <3-4>

Do you want the synthesizer generated with 3 or 4 voices? Music arranged for four voices can be played on a 3-voice system. The compiler simply ignores the fourth
voice

For slow systems (less than 4 Mhz) three voices will sound better than four. Three voice systems also require less memory.

At this point SMS will generate the synthesizer and the note/frequency tables. It may take a second or two.

The next three steps will setup the input port address and test the switch register on the interface board.

STEP 4: TEMPO (INPUT) PORT? <00-FF>

What is the address of the INPUT port SMS will use to read the switch register on the inferface board? Enter a two digit hexadecimal number.

STEP 5: INVERT? <Y/N>

Should SMS invert the data before using it? If you don't know how to answer this question, type a Y. The correct answer will be determined in the next step.

STEP 6: TEST READ? <Y/N/A>

This is a diagnostic step. If you answer Y, SMS will read the switch register and display the status of each of the bits.
If you answer N, SMS will proceed to the next step.
If you answer A, SMS will return to step 4 so you can alter your answers to those questions.

Set all the switches in the down or off position and answer Y. SMS should display eight 0's. If it displays eight 1's, you should type an A and alter your answer at step 5. Repeat this step with different switch settings to verify the correct reading of each bit. The left-most bit displayed should correspond to the left-most switch, etc. If the display does not change when you change the switches, double check the input port connection and address. If the display changes but the bits do not match the switches, double check the jumpers on the interface board.

The next three steps will setup the output port address and test the resistor network on the interface board. All the switches should be in the off position, i.e.,the register should be all 0's.

STEP 7: MUSIC (OUTPUT) PORT? <00-FF>

What is the address of the OUTPUT port. Enter a two digit hexadecimal number.

STEP 8: TEST TONE? <Y/N/A>

This is a diagnostic test, if you answer Y, SMS will generate a 440 hertz sine wave.

If you answer N, SMS will proceed to the next step.

If you answer A, SMS will return to step 7 so you can alter your answer to that question.

Answer Y and check the quality of the test tone. It will last several seconds. The tone should be smooth and pure like a tuning fork. If you don't hear anything, double check the output port connection and address. If the tone is distorted, double check the jumpers on the interface board.

STEP 9: TEST SCALE? <Y/N>

If you answer Y, SMS will play a sample scale with each of the four different tone colors (registers A, B, C and D.)

If you answer N, SMS will proceed to the next step.

The next five steps allow you to modify the wave forms of the different registers. If you are not already familiar with the use of the system, answer the next step N.

STEP 10: ALTER SINE TABLES? <Y/N>

If you answer Y, SMS will proceed to the next step where you will have no choice but to define a new register.

If you answer N, SMS will execute the last step in the system generation proceedure. The last step is version dependent so proceed to the section that applies to your system - CP/M, DOS or SOLOS/CUTER.

STEP 11: REGISTER NAME? <A-D>

Enter the name of the register you wish to modify.

STEP 12: NUMBER OF PARTIALS? <1-16>

The wave form of a register is defined by a sum of sine waves or partials. The frequency of each partial is an integer multiple of the fundamental frequency.

Partial #1 is the fundamental, partial #2 is the first harmonic and is two times the fundamental frequency, partial #3 is the second harmonic and is three times the fundamental, etc. Enter the number of the highest partial you wish to define.

Unless you are after special effects, the number of the highest partial should not exceed two times the CPU clock rate in mega-hertz (rounded-up), i.e., 4 partials for 2 Mhz, 8 partials for 4 Mhz, etc. This will keep aliasing to a minimum. Refer to the How It Works section for a complete explanation.

STEP 13: PARTIAL #01, WEIGHTING FACTOR? <0-255>

This step will be repeated for each partial to be defined. Enter a number that represents the relative strength of the named partial. A partial with a weight of 200 will be twice as prominent (twice as loud) as a partial with a weight of 100. A partial with a weight of 0 makes no contribution to the final wave form. All weights are normalized so that a register defined with all partials of weight 1 will be exactly the same as a register defined with all partials of weight 255.

STEP 14: VOLUME? <1-256>

Enter the overall amplitude of the register. The higher the number the louder it will be.

SMS will now generate the register you have just defined. When it has finished the calculations (it may take several seconds)
SMS will return to STEP 9 to demonstrate the results.

LAST STEP - CP/M

STEP 15: SAVE CONFIGURED SOFTWARE MUSIC SYNTHESIZER PROGRAM

The configured SMS program is now in memory and must be saved on disk before it can be used. This is the one and only time the configured program will be in a state that can be saved. Note how many 256-byte blocks the program takes; you will use this number in the SAVE command. Remove the SMS diskette, place the CP/M system diskette in drive A and hit any key.

Choose a name for the finished program, such as, STDMS4 if you generated the standard registers with a 4 voice synthesizer. DO NOT call it SMS as that name should be reserved for the system generation program.
SAVE the program with a file type of .COM.

SAVE 16 STDMS4.COM

If you have not used the SMS system before, you are probably anxious to hear what it can do. Type the name of the program you just saved to reload and run it.

STDMS4

After you see the copyright notice return the SMS diskette to drive A and type

GET LONE

to get and play the music called Lone. In a few seconds you'll be on your way. Hi-yo Silver, away!

LAST STEP - DOS

STEP 15: SAVE CONFIGURED SOFTWARE MUSIC SYNTHESIZER PROGRAM

The configured SMS program is now in memory and should be saved on disk before it is used. This is the one and only time the configured program will be in a state that can be saved.

Choose a name for the finished program, such as, STDMS4 if you generated the standard registers with a 4 voice synthesizer. DO NOT call it SMS as that name should be reserved for the system generation program.

Note how many 256-byte blocks the program takes and create a directory entry for it and save the program.

```
CR STDMS4 16
SF STDMS4 2D00
TY STDMS4 1 2D00
```

If you have not used the SMS system before, you are probably anxious to hear what it can do. Type JP 2D00 to execute the program you just saved. After you see the copyright notice type

GET LONE

to load and play the music called Lone. In a few seconds you'll be on your way. Hi-yo Silver, away!

LAST STEP - SOLOS/CUTER

STEP 15: SAVE CONFIGURED SOFTWARE MUSIC SYNTHESIZER PROGRAM

The configured SMS program is now in memory and should be saved on tape before it is used. This is the one and only time the configured program will be in a state that can be saved. Remove the SMS cassette and place a blank tape in the machine. Choose a name for the finished program, such as, STDS4 if you generated the standard registers with a 4 voice synthesizer. DO NOT call it SMS as that name should be reserved for the system generation program. Note the program starting and ending addresses and transfer it to the cassette.

SAVE STDS4 100 1022

Rewind, remove and label that cassette and place the SMS cassette in the machine again.

If you have not used the SMS system before, you are probably anxious to hear what it can do. Type

EXEC 100

to execute the program you just saved. After you see the copyright notice type

GET LONE

to load and play the music called Lone. In a few seconds you'll be on your way. Hi-yo Silver, away!

USING THE SYSTEM

KEYBOARD INPUT

SMS will accept either upper or lower case characters. Lower case input will be translated to upper case before it is echoed on the display device and stored in the 80 byte command buffer. All input lines are terminated with a carriage return or enter key.

The following characters have special functions.

DEL (delete) or RUB (rubout)
Deletes the last character in the current input buffer. CP/M version echoes the character deleted. DOS and SOLOS/CUTER echo a '_' to backspace the display device.

_(underline)
DOS only: same as DEL.

cntrl/C
CP/M and DOS only: Cancels current operation in progress, if any, and deletes entire command line.

MODE (cntrl/@)
SOLOS/CUTER only: Cancels current operation in progress, if any, and deletes entire command line.

LF (line feed)
Pre-types next EDIT line number. Active only at the beginning of a new line.

KEYWORDS

All system commands consist of a keyword optionally followed by one or more operands. Operands are separated from each other and from the keyword by one or more spaces. The keyword may be abbreviated (only the first character is significant) but may not contain any leading or embedded spaces.

In the following description:
'linel' and 'line2' are one to four digit decimal line numbers.
'addrl' and 'addr2' are one to four digit hexadecimal addresses.
'filel', 'file2', etc. are file names conforming to the file naming conventions of the host operating system. They may be preceeded or followed by an alternate drive number, if necessary.
'xx' is any valid part number.

DELETE linel line2

Deletes lines of text from the current text file from linel to line2, inclusive. If line2 is omitted or is less than linel, only linel, if it exists, is deleted. If Linel is omitted, no lines are deleted.

LIST linel line2

Lists lines of text from the current text file from linel to line2, inclusive. If line2 is omitted or is less than linel, only linel, if it exists, is listed. If linel is omitted, the entire text file will be listed.

The listing may be temporarily stopped by pressing the space bar (CP/M and SOLOS/CUTER versions only). Press any key to continue the listing.

The listing may be terminated at any time by the 'line delete' key. (MODE or cntrl/@ for SOLOS/CUTER; cntrl/C for CP/M and DOS.)

NEW addrl

The current text file, if any, is lost and a new text file is created at addrl. If addrl is omitted, the file is created at the lowest address in the available workspace. The beginning and ending address of the new file are displayed.

SMS always performs a NEW function when it is first loaded into memory and executed.

FILE addrl

Validates the integrity of the text file at addrl. If addrl is omitted, the current text file is validated.

If the validation is successful, the beginning and ending addresses of the file are displayed. Otherwise, an error message is displayed.

READ filel

Performs an internal NEW and attempts to read filel into memory. If the read operation is successful, the FILE function is performed to validate the file and establish the current file pointers. An error message will be displayed if: the file cannot be found; an I/O error occurred; the file is too large for the available workspace; the file cannot be validated.

WRITE filel

The current text file is written as filel. An error message will be displayed if: an I/O error occurred; (DOS and CP/M only) the disk or directory is full; (DOS only) the file already exists but is too small to contain the current text file.

CP/M only: WRITE performs these BDOS functions; ERASE filel, CREATE filel, WRITE filel, CLOSE filel. If the WRITE operation fails, filel, if it previously existed, will be lost.

DOS only: WRITE will attempt to rewrite filel if it already exists, otherwise, filel will be created before writing.

SCORE addr1

The current text file is compiled and the resulting object code is placed in memory starting at addr1. If addr1 is omitted, the object code is placed at the next available location (after the text file) in the work space. After a successful compilation, the beginning and ending addresses of the object code are displayed.

An error message will be displayed if the compilation terminates abnormally.

There are no restrictions on addr1, other than it be within the allocated workspace.

OVERLAP

Moves the current text file to the highest possible address in the work space and SCORES it to the lowest possible address. This function is useful when compiling large files as the end of the object code can OVERLAP the beginning of the source code.

DO NOT OVERLAP a text file that has not been saved on disk or tape. This function implies the probable destruction of the current text file. Do not expect to be able to use it in any way once this command is given.

PLAY xx

Directs the synthesizer to PLAY the most recently SCORED object code starting with PART xx. If xx is omitted, the entire piece is played. An error message will be displayed if the PART cannot be found or if the text file was not recently SCORED or OVERLAPPED.

The switch register on the interface board is used to control the operation of the synthesizer while a piece is being PLAYED. Switches 1-7 control the TEMPO.

Switch #8, when ON (1), will stop the synthesizer.

GET file1 file2 file3 ....

This is a repetitive function. It will perform a READ, OVERLAP and PLAY for each of the files named in the operand.

Switch #8 or any error will terminate the function.

QUIT

Returns control to the host operating system.

CP/M only: Performs a WARM BOOT. Be sure drive A contains a disk with a valid copy of CP/M on it before using this command.

DOS and SOLOS/CUTER only: It is possible to re-enter SMS after it has been QUIT. JP 2D00 or EXEC 100, provided intervening operations have left it undisturbed. After re-entering, issue the FILE command to re-establish the current text file pointers.

TEXT EDITOR

Any line beginning with a decimal digit is passed to the editor for insertion in the current text file. All lines in the text file begin with a one to four digit decimal line number followed by at least one space. The numbered text lines are maintained in numeric sequence.

The automatic line number function of the LF (line feed) key can be used when entering a file into the system for the first time. When the LF key is the first key entered on a new line, SMS will pre-type a line number that is 10 larger than the last line entered. Enter the first line manually to get the automatic numbers started in the right place.

0000 is not a valid line number.

The editor has no real 'editing' capability. To modify an existing line it is necessary to retype and enter the entire line. This limitation can be partially overcome by keeping the text lines as short as practical. This will reduce the amount of re-typing neccessary should a line need to be modified. Another possibility is to use a more advanced text editor to create and maintain SMS source files.

DOS and SOLOS/CUTER users who also have access to Processor Technology's ALS-8 may find it more convenient to use the full screen edit feature of that system. The SMS file structure is compatible with ALS-8. Always issue a FILE command when re-entering SMS. Likewise, issue an FCHK command when re-entering ALS-8 to update its current file pointers.

CP/M users who would rather use a more powerful, disk-based editor will find the two file conversion programs supplied on the SMS diskette useful. SMSTOMUS will convert an SMS file to CP/M ED format. In the course of conversion the line numbers are stripped off and the file type is changed from .SMS to .MUS. The twin program MUSTOSMS converts by adding line numbers to a file of type .MUS and creating the equivalent SMS type file., e.g.;

```
A>SMSTOMUS prog B:
          Converts A:PROG.SMS to B:PROG.MUS

A>MUSTOSMS B:prog
          Converts B:PROG.MUS to A:PROG.SMS
```

MUSIC LANGUAGE DEFINITION

The music language provides the means by which standard musical notation is transcribed into a symbolic form usable by the computer.

The use of the language can be shown best by example. Figure #1 is an excerpt from J. S. Bach's Capriccio. Figure #2 is the same music transcribed for a three voice synthesizer. While the two figures may appear to have little in common, they both contain essentially the same information.

In Figure #2 each line begins with a four digit number followed by a space. These line numbers are required by the File Editor and have no bearing on the music. The compiler will ignore all characters on a line up to the first space. In the fol lowing discussion the lines in Figure #2 will be referenced by line number. Remember, the contents of each line effectively begin after the first space.

Lines 10, 20 and 30 are informational only. A slash (/) appearing anywhere in a line causes the remainder of that line to be ignored.

For the purposes of transcription, a piece of music is considered to be subdivided into parts, measures and voices. A part defines one or more measures that are played in the same key and tempo and usually correspond to a portion of the piece that may be repeated, i.e., a phrase, stanza, chorus, etc. Parts are defined by the letter 'P' followed by a two digit, decimal number.

Measures are indicated by a character string beginning with the letter 'M' and ending with a space. The characters following the 'M', usually a measure number, are ignored by the compiler and only serve as a reference between the printed musical score and the transcribed music language text.

A voice is a separate 'strand' of music, in harmony or counter point. A trio has three voices and a quartet four. Up to four voices may be defined and they are identified by V1, V2, V3 and V4. As a convenience, each measure begins with an implied V1. Music arranged for a four voices can be played by a three voice synthesizer except Voice 4 will be silent.

In the example, the Aria consists of two repeated sections, one 5 measures long and the other 7 measures long. The first section is defined in line 40 as 'P50'. The choice of the two digit number is arbitrary and serves only to identify the part. However, each part defined must have a unique number.

Line 200 defines Part 51 as being a repeat of Part 50. Likewise, Part 53 (line 390) is a repeat of Part 52 (line 210).

Line 50 defines the key signature. You need only specify the number of sharps (#) or flats (&); the compiler will know which notes are affected. If the key signature is not specified, C-major (no sharps or flats) is assumed.

Line 60 defines the time signature and tempo. 'NQ' indicates that each quarter note gets a beat (H=half, Q=quarter, I=eighth, S=sixteenth). '=C0' indicates the relative length of a beat. Normally, the tempo parameter is determined experimentally using the switch register on the interface board. It may also be necessary to alter the time signature parameter to achieve to desired tempo.

Line 62 defines transposition. The character '<' or '>' followed by a number defines the direction and number of semi-tones the piece is to be transposed (<=down, >=up). It is common practice to transpose a piece of music as it is transcribed to accomidate the range and fingering of the new instrument. While SMS should not pose any 'fingering' problems, it is often desirable to transpose a piece down a few semi-tones to avoid the distortion and aliasing present in the higher notes. This is especially true with 2 Mhz, 4-voice synthesizers.

Line 64 defines the tone color registers to be used by the different voices. Voice 1 uses register C (clarinet), voice 2 and 3 use register B (oboe) assuming, of course, the registers have not been altered during the configuration phase. The default register is D (organ).

The music proper begins in line 70 with the definition of measure 1.

Each of the symbols representing a 'note' in standard musical notation imply two pieces of information:

1. Its shape, along with the time signature, defines how long, relative to a beat, the note is to be held.

2. Its position on the staff, along with the key signature and clef define the note to be played.

Transcribing this two dimensional form into a single line requires two characters to represent each note. The time value, or shape, is defined by a single letter: W=whole, H=half, Q=quarter, I=eighth, S=sixteenth, T=thirty-second, X=sixty-fourth. The time value may be modified by additional symbols. Dotted notes are indicated by a period (.) following the letter. Triplet time values are indicated by a colon (:) following the letter, e.g. 'Q' means all the notes following are quarter notes; 'I.' means all the notes following are dotted eight notes; 'S:' means all the notes following are sixteenth note triplets. Note length modifiers may be combined: 'H:..'means all the notes following are double dotted half note triplets. (Dot modifiers cannot precede a triplet modifer, i.e. 'Q.:' is not valid.)

The staff position of a note is defined by its relationship to a fixed point on the staff, Middle-C. Middle-C is always location 0 and notes above it are defined by a positive (+) displacement while notes below it are defined by a negative (-) displacement. See Figure #3. The displacement uses a hexadecimal-like number scale with the letters A to G representing the numbers 10 to 16. The maximum positive displacement is +G, the maximum negative displacement is -F. A dollar sign ($) defines a rest.

Coding note displacements can be simplified by specifing a default displacement sign or clef. An asterisk (*) sets the default to '+' or treble clef, and all unsigned notes following are assumed to be '+'. An at-sign (@) sets the default to '-' or bass clef, and all unsigned notes following are assumed to be '-'.

Accidentals are indicated with a sharp (#), flat (&) or natural (%) sign immediately following the note affected. Accidentals stay in effect until the end of the measure or are modified by another accidental. Double sharps (##) and flats (&&) and natural sharps (#% or %#) and flats (&% or %&) are allowed. Double naturals (%%) and flatted sharps (#& or &#) will give unpredictable results and should be avoided.

You should now be able to decipher all the symbols in lines 70, 80 and 90 and relate them to the first measure in Figure #1.

Line 100 introduces several new symbols. The paranthesis indicate reiterative compilation. The number following the right paranthesis is the reiteration count which tells the compiler how many more times to compile the bounded sequence. Line 100 will be compiled as if it were written:

```
M2 *I6;SD6"I6;SD6"I6;SD6"I6;SD6"
```

Reiteration should not be confused with Part repetition. Reiteration occurs at compile time; Part repetition directs the synthesizer to re-play a particular part. Reiteration blocks may not be nested.

Notes may be further modified by 'expression' modifiers which alter the way a note is played, but not its pitch or overall duration, by replacing a portion of the note with a rest. There are two major types of expression, staccato and articulation, and each type has two forms, long and short. Short staccato, indicated by a comma (,), shortens the note by 1/2 and adds a rest equal to 1/2 the notes duration. Long staccato, indicated by a semi-colon (;), is similar except the note is shortened by 1/4 of its value and adds a 1/4 rest. Articulation is used to introduce a small rest after a note to separate it from the note following. Short articulation, indicated by an apostrophe ('), shortens the note by an amount equal to 1/3 of a 1/128 note and adds a rest of that value. Long articulation, indicated by a quote ("), is similar except the amount shortened is double, 2/3 of a 1/128 note. In all cases, expression modifiers affect only the note they follow. Accidentals must precede expression modifiers.

Measure 5 (lines 170-190) presents a very common problem; too many voices. An experienced musician will probably have no trouble. For the rest of us, it will be a matter of blind luck or painstaking trial and error deciding which notes to use and which to ignore. Ultimately, the 'right' solution is the one that sounds best.

Thus far, the assumption has been that the music being transcribed is in bass/treble clef notation. To take advantage of music arranged for different instruments in different clefs (soprano, alto, tenor, etc.), each voice can be defined as belonging to a different clef. Voice/clef definition is indicated by an up-arrow (^) followed by a number that is the displacement from Middle-C in the clef being defined to Middle-C in the treble clef. For example,

```
V1^-2 V2^-6 V3^-8 V4^-C
```

defines Voices 1-4 as soprano, alto, tenor and bass, respectively. Because all clefs are defined relative to the treble clef, each voice is transcribed exactly as if it were the treble clef, and no clef symbols (* or @) need be used.

Another useful transposition would be V4^-7 to lower Voice 4 one octave to get a low bass sound.

Often, when transcribing part-music, the accidentals in one voice interfere with the accidentals in another voice. This is because the compiler applies every accidental to all voices. Option 1, specified O1, will limit the application of accidentals to the voice in which they appear.

MUSIC LANGUAGE SUMMARY

After discarding line numbers, the Music Language is processed in a single, free format, string of symbol groups. All spaces and line boundaries between symbol groups are ignored. A symbol group consists of a single character symbol optionally followed by one or more symbol modifiers.

Unless otherwise specified, all numeric data is hexadecimal.

NOTE SYMBOL	NAME	STAFF POSITION (see TABLE #3)
+G	E	
+F	D	
+E	C	
+D	B	
+C	A	
+B	G	
+A	F	
+9	E	
+8	D	
+7	C	
+6	B	
+5	A	
+4	G	
+3	F	
+2	E	
+1	D	
0	C	Middle-C
-1	B	
-2	A	
-3	G	
-4	F	
-5	E	
-6	D	
-7	C	
-8	B	
-9	A	
-A	G	
-B	F	
-C	E	
-D	D	
-E	C	
-F	B	
$	REST	(any)

NOTE MODIFIER	MUSICAL EXAMPLE	NAME
#		ACCIDENTAL SHARP
&		ACCIDENTAL FLAT
%		ACCIDENTAL NATURAL
##		DOUBLE SHARP
&&		DOUBLE FLAT
%#		NATURAL SHARP
%&		NATURAL FLAT

EXPRESSION MODIFIER	MUSICAL EXAMPLE	NAME
'		SHORT STACCATO
;		LONG STACCATO
'	(none)	SHORT ARTICULATION
"	(none)	LONG ARTICULATION

TIME VALUE SYMBOL	MUSICAL EQUIVALENT	NAME OF NOTE
W		WHOLE note
H		HALF note
Q		QUARTER note
I		EIGHT note
S		SIXTEENTH note
T		THIRTY-SECOND note
X		SIXTY-FOURTH note

TIME VALUE MODIFIER	MUSICAL EQUIVALENT	NAME
.		DOTTED note
:		TRIPLET

SYMBOL	MODIFIER	DEFINITION
/	(none)	Comments. The remainder of the line is ignored.
P	2-digit decimal	Defines the beginning of the PART named by the modifier. The current MEASURE is ended. The current PART is ended.
R	2-digit decimal	REPEAT the PART named by the modifer. The named PART must be previously defined. This symbol group should be preceded by a PART group and followed by a TEMPO group and/or REGISTER group.
M	any string and a space	Define the beginning of a MEASURE. The current MEASURE is ended. ACCIDENTALS are dropped, the KEY SIGNATURE is restored.
V	1,2,3 or 4	All the NOTES following are added to the previous NOTES of this MEASURE belonging to the VOICE named by the modifer.
<	hex digit	All the NOTES in the current and following MEASURES are transposed DOWN the number of semi-tones specified by the modifer.
>	hex digit	All the NOTES in the current and following MEASURES are transposed UP the number of semi-tones specified by the modifier.
*	(none)	Unless otherwise indicated, all the NOTES following are assumed to be '+' or TREBLE clef.
@	(none)	Unless otherwise indicated, all the NOTES following are assumed to be '-' or BASS clef.
^	signed hex	TRANSPOSE all the NOTES following that belong to the current VOICE up or down the number of steps indicated by the modifier.
K	digit 0-7 and # or &	Define KEY SIGNATURE by number and type specified by the modifier.
N	H,Q,I,S or T	Define TIME SIGNATURE, set NOTE type in

		modifer to one BEAT.
=	2-digit hex	Set TEMPO to value of modifer.
Y	A,B,C or D	Set the current VOICE in the current PART to the REGISTER specified by the modifier.
(	(none)	Begin REITERATION.
)	hex digit	REITERATE the number of times specified by the modifier.
O	hex digit	Set the OPTIONS to the value of the modifier.

ERROR MESSAGES

When an error is detected an error number will be displayed and, whenever possible, the current command or text line will be displayed with a question mark inserted at or near the place where the error was encountered.

ERR 1: ADDRESS OVERFLOW

SMS attempted to write to a memory location that was not within its allocated workspace.

1) The operand of a NEW, FILE or SCORE command is too large or too small. An operand with a value of zero is treated as an omitted operand. Correct and re-enter the command.

2) There is not enough memory to SCORE the current file. Use the OVERLAP command after saving the current file.

3) There is not enough memory to EDIT the current file. Use the FILE command to verify the size of the file and its starting address. If the file is in high memory as a result of a previous GET or OVERLAP command,and read it back into low memory. If it is already in low memory, consider expanding your system or splitting the file into two or more smaller sections.

4) There is not enough memory to READ or GET to requested file. Expand the memory in your system.

5) The line number contains more than four digits. Correct and re-enter the line.

ERR 2: MEASURE OVERFLOW

The measure being compiled contains more than 64 notes per voice. Solution: Split the measure in half. Carry forward any accidentals.

Note: Staccatto, pizzicato and articulation generate a note and a rest so count them as two notes.

ERR 3: INVALID FILE FORMAT

The file in memory cannot be processed by SMS. If this error follows a READ or GET, the requested file was probably not originally created by SMS or it has somehow been corrupted. If this error results from the FILE command, the file at addr1 or the current file is invalid. Remember, OVERLAP and GET may destroy the current file.

ERR 4: SYMBOL OUT OF CONTEXT

1) The command operand contains an invalid hexadecimal digit.
2) The line number does not end with a space or it contains an invalid digit.
3) The TEMPO symbol '=' is not followed by two hexadecimal digits.
4) The compiler was expecting a hexadecimal digit or a note symbol (0-9 or A-G) but none was found.

ERR 3: INVALID FILE FORMAT

The file in memory cannot be processed by SMS. If this error follows a READ or GET, the requested file was probably not originally created by SMS or it has somehow been corrupted. If this error results from the FILE command, the file at addr1 or the current file is invalid. Remember, OVERLAP and GET may destroy the current file.

ERR 4: SYMBOL OUT OF CONTEXT

1) The command operand contains an invalid hexadecimal digit.
2) The line number does not end with a space or it contains an invalid digit.
3) The TEMPO symbol '=' is not followed by two hexadecimal digits.
4) The compiler was expecting a hexadecimal digit or a note symbol (0-9 or A-G) but none was found.

ERR 5: PARAMETER ERROR

1) The number of sharps or flats in the KEY signature is too large or is not followed by SHARP '#' or FLAT '&' symbol.
2) The note value in the TIME signature is not H, Q, I, S or T.
3) The VOICE specified is not 1, 2, 3 or 4.
4) The REGISTER specified is not A, B, C or D.

ERR 6: INVALID PART NUMBER

1) The PART or REPEAT specified is not a single alpha character or is not a decimal digit followed by a hexadecimal digit. 00 is not a valid PART number.
2) The PART is already defined.
3) The REPEAT is not previously defined.
4) The piece cannot be PLAYED because the PART cannot be found or because the piece needs to be SCORED.

ERR 7: NOTE OVERFLOW

Dotted whole notes, some 7-dotted triplets and other unusual combinations cannot be compiled.

ERR 8: FILE I/O ERROR

The host operating system reported an I/O or other error during the last file operation.
1) READ or GET: File not found.
2) WRITE: Disk or directory full.
3) READ, WRITE or GET: Unrecoverable I/O error.

COMPATABILITY

The SMS compiler syntax was designed to be compatible with the now unavailable STC Music System (STCMS). Other than that, the two programs have nothing in common. Music originally arranged for STCMS may require some changes to be playable by SMS.

A major difference between the two systems is the function of the TIME SIGNATURE. Use the following table to change the 'N' parameter.

STCMS	SMSS
NH	NT
NQ	NS
NI	NI
NS	NQ
NT	NH

The TEMPO parameter '=' must also be adjusted but there is no direct correspondence. Use the switch register to establish the correct tempo and then transfer it to the music source.

The order of dot and triplet modifers is reversed.

STCMS	SMSS
Q.:	Q:.

The compiler will accept either 2-digit or single letter part numbers. It is possible that SMSS will play the parts of a piece originally arranged for STCMS in the wrong order. This can happen because SMSS plays all the parts in the order defined, whereas STCMS always plays the parts in alphabetic order.

STANDARD REGISTERS

The standard tone color registers are based on the spectral analysis of orchestral instruments and are generated with the following recipes: (Systems running with a clock rate of less than 3.56 Mhz do not generate partials 5 through 7.)

	Partial/Weighting Factor								
Register	**1**	2	**3**	4	**5**	6	7	8	**Vol**
A-trumpet	**224**	240	**240**	160	**80**	64	**48**	48	**224**
B-oboe	**64**	128	**240**	128	**240**	32	**16**	16	**240**
C-clarinet	**224**	0	**80**	0	**240**	0	**48**	80	**160**
D-organ	**240**	64	**0**	128	**0**	0	**0**	32	**176**

HOW IT WORKS

The SMS synthesizer uses a sampling technique commonly used in professional digital synthesizers.

The Sampling Theorem tells us that any wave form, no matter how complex, can be reconstructed from a rapid succession of discreet voltages. The reconstructed wave form will actually be a stepped approximation of the original but if the steps are small enough they will not be noticed. The size of the steps is determined by the sample rate and the frequency of the wave being repoduced.

Consider one cycle of a simple sine wave. If we were to measure its amplitude at 100 equally spaced intervals and record the values in a table we could reproduce it by serially setting a voltage equal to each entry in the table. Since the table represents exactly one cycle, the frequency of the synthesized wave will be equal to the sample rate divided by 100 (the number of entries in the table). At 1000 samples per second the entire table will be accessed 10 times per second, producing a 10 hertz sine wave. Doubling the sample rate will double the frequency of the synthesized sine wave.

Using the same table and sample rate we could effectively double the frequency of the synthesized wave by accessing every other table entry. This gets us through the entire table twice as fast. Similarly, taking every third or fourth table entry will triple or quadruple the frequency. This is the basis of the SMS synthesizer, except to get musically accurate frequencies, it is necessary to skip a fractional number of table entries.

Theoretically, the highest frequency that can be synthesized is equal to half the sample frequency. This is called the Nyquist frequency. As this limit is approached, the synthesized wave form becomes more and more distorted since there are fewer samples in each cycle. As the distortion increases, a secondary tone, or alias, is produced. If the Nyquist frequency is exceeded, the synthesized frequency is replaced by its alias. This phenomenon is not limited to the fundamental frequency. It affects each component of the synthesized wave form.

A certain amount of distortion is unavoidable and the SMS synthesizer has its share. Ideally, a low pass filter with a sharp cutoff at the Nyquist frequency will eliminate most of the high frequency distortion components. Such a filter on the SMS interface board was considered impractical because the cutoff frequency required depends on knowing the sample rate of the synthesizer, which depends on the CPU clock rate. Instead, the interface board employs a single stage, high frequency roll-off.

Additional high frequency suppression should be done with the amplifier's tone controls.

For those users interested in making their own low pass filter, the synthesizer's Nyquist frequency can be calculated by:

$$F = 500 * C / S$$

where F is the Nyquist frequency in kilo-hertz, C is the micro processor clock rate in mega-hertz and S is the number of CPU cycles per sample. S=277 for a three voice synthesizer; S=358 for a four voice synthesizer.

Aliasing cannot be filtered out, it can only be avoided. If aliasing is a problem, redefine the registers with fewer partials or transpose the entire piece down a few semi-tones. An alternative solution would be to speed up your CPU.

SWITCH REGISTER TO TEMPO CONVERSION TABLE

The switch register on the interface board is used to set or alter the tempo of a piece while it is playing. When all the switches are in the OFF, or 0, position the piece will play at the tempo defined when it was compiled. Otherwise, the hexadecimal value of the switch register is used to set the tempo. The following table shows the hexadecimal and decimal value for each of the switch settings.

Once the correct tempo has been established by experimentation with the switch register, the hexadecimal value should be transfered to the source file. Tempo settings below 80 (hex) may cause an undesirable shift in frequency and should be avoided. It may be necessary to adjust the time signature parameter to get the tempo between 80 and FE. Increasing the note value of the time signature will allow you to double the value of the tempo. e.g. NQ=C0 is preferable to NI=60.

The least significant bit (the right-most switch) is used as a stop switch. When that switch is ON, or 1, the synsthesizer will stop and SMS will return to the command mode.

SWITCH	HEX	DEC	SWITCH	HEX	DEC	SWITCH	HEX	DEC
11111110	FE	254	10111110	BE	190	01111110	7E	126
11111100	FC	252	10111100	BC	188	01111100	7C	124
11111010	FA	250	10111010	BA	186	01111010	7A	122
11111000	F8	248	10111000	B8	184	01111000	78	120
11110110	F6	246	10110110	B6	182	01110110	76	118
11110100	F4	244	10110100	B4	180	01110100	74	116
11110010	F2	242	10110010	B2	178	01110010	72	114
11110000	F0	240	10110000	B0	176	01110000	70	112
11101110	EE	238	10101110	AE	174	01101110	6E	110
11101100	EC	236	10101100	AC	172	01101100	6C	108
11101010	EA	234	10101010	AA	170	01101010	6A	106
11101000	E8	232	10101000	A8	168	01101000	68	104
11100110	E6	230	10100110	A6	166	01100110	66	102
11100100	E4	228	10100100	A4	164	01100100	64	100
11100010	E2	226	10100010	A2	162	01100010	62	98
11100000	E0	224	10100000	A0	160	01100000	60	96
11011110	DE	222	10011110	9E	158	01011110	5E	94
11011100	DC	220	10011100	9C	156	01011100	5C	92
11011010	DA	218	10011010	9A	154	01011010	5A	90
11011000	D8	216	10011000	98	152	01011000	58	88
11010110	D6	214	10010110	96	150	01010110	56	86
11010100	D4	212	10010100	94	148	01010100	54	84
11010010	D2	210	10010010	92	146	01010010	52	82
11010000	D0	208	10010000	90	144	01010000	50	80
11001110	CE	206	10001110	8E	142	01001110	4E	78
11001100	CC	204	10001100	8C	140	01001100	4C	76
11001010	CA	202	10001010	8A	138	01001010	4A	74
11001000	C8	200	10001000	88	136	01001000	48	72
11000110	C6	198	10000110	86	134	01000110	46	70
11000100	C4	196	10000100	84	132	01000100	44	68
11000010	C2	194	10000010	82	130	01000010	42	66
11000000	C0	192	10000000	80	128	01000000	40	64

FIGURE #1

Nº 3. Capriccio

sopra la lontananza del suo fratello dilettissimo.

Ist eine Schmeichelung der Freunde, um denselben von seiner Reise abzuhalten.

Aria di Postiglione.

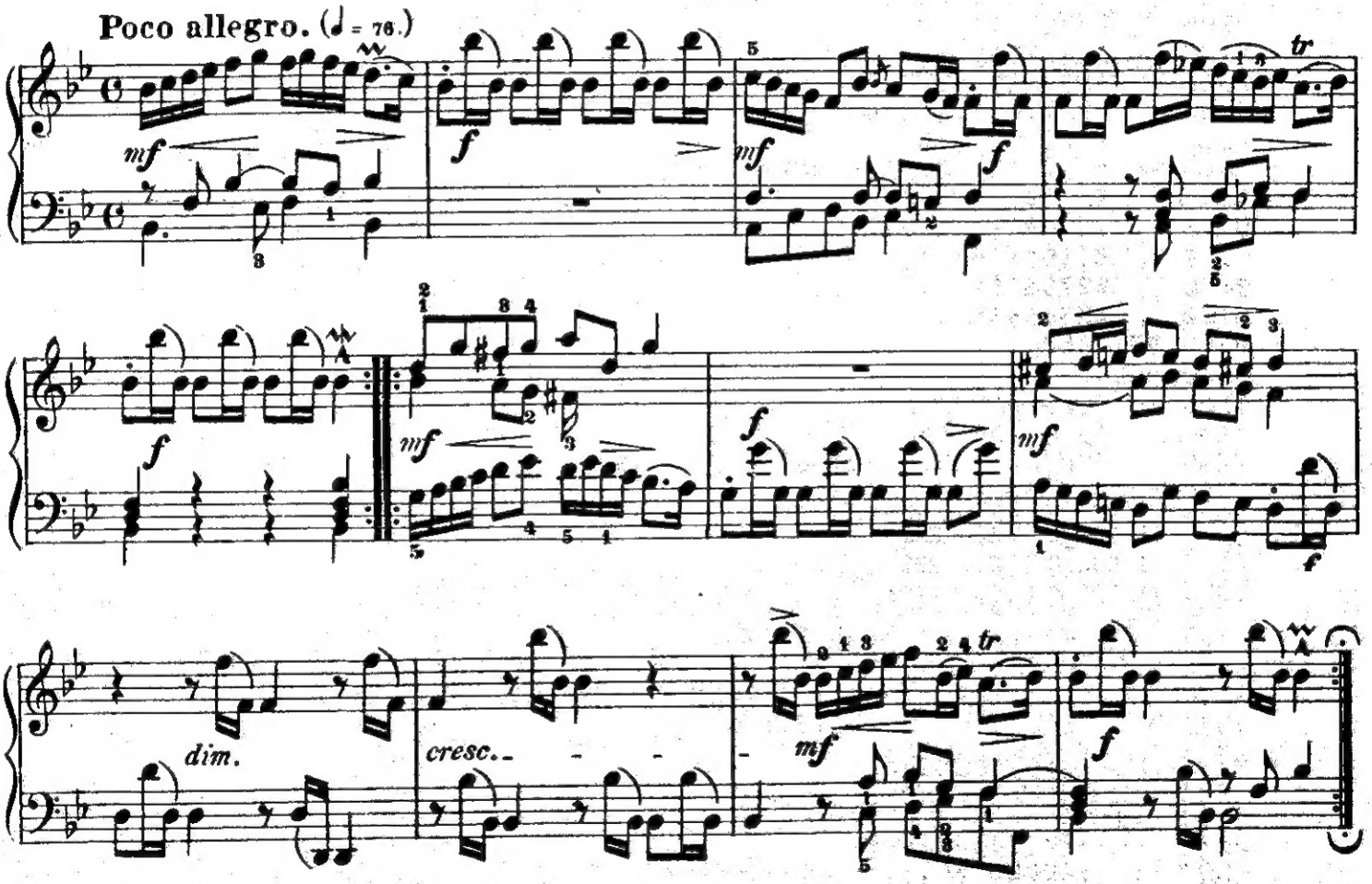
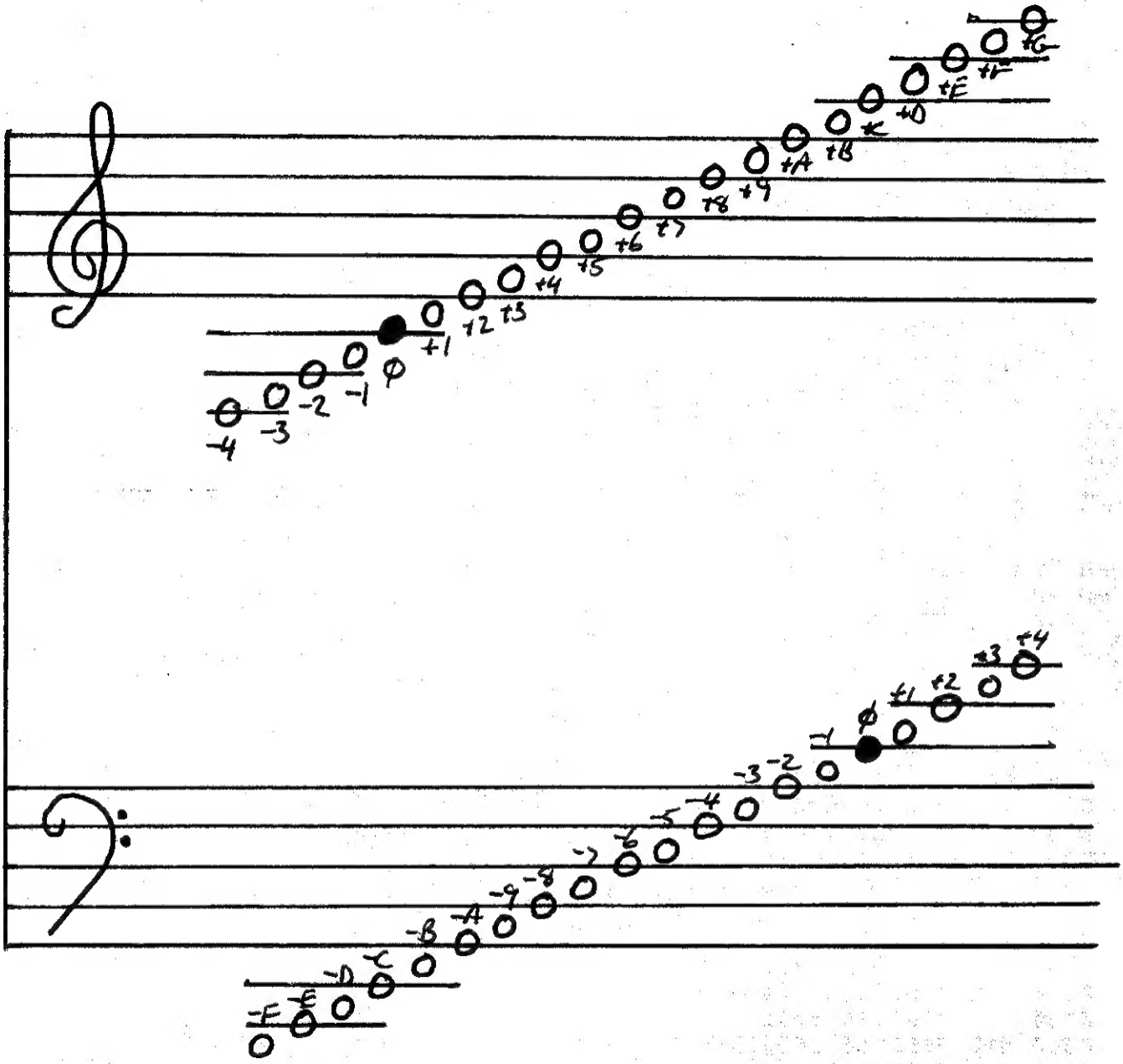


FIGURE #2

```
0010 / CAPRICCIO
0020 / SOPRA LA LONTANANZA DEL SUO FRATELLO DILETTISSIMO
0030 / J.S. BACH
0040 P50 / ARIA DI POSTIGLIONE
0050 K2&
0060 NQ=E0 /POCO ALLEGRO
0062 <2
0064 V1YC V2YB V3YB
0070 M1 *S6789IABSABA9I.8S7
0080     V2@I$4Q1I12Q1
0090     V3@Q.8I5Q48
0100 M2 *(I6;SD6")3
0110 M3 *S7654I365S43"I3;SA3"
0120     V2@Q.4"I445%Q4
0130     V3@I9768Q7B
0140 M4 *I3;SA3"I3;SA9&8767I.5S6"
0150     V2@Q$I$4"43Q4"
0160     V3@Q$I$785&Q4
0170 M5 *(I6;SD6")3
0180     V2@Q4$$4
0190     V3@Q6$$6
0200 P51 R50
0210 P52
0220 M6 *I8BA#BC8QB
0230     V2*Q6I54S3#
0240     V3@S3210*I12S1210@I.1S2
0250 M7 V3@(I3;S+43")2I3;+4
0260 M8 *I7#S89%IA987#Q8
0270     V2*Q5I5654Q3
0280     V3@S2345%I63456;S+16"
0290 M9 *Q$I$SA3"Q3I$SA3"
0300     V2@I6;S+16"Q6I$S6D"ID
0310 M10 *Q3I$SD6"Q6
0320      V2@I$S18"Q8I$S18"I8;S18"
0330 M11 *I$SD6"6789IAS67I.5S6"
0340      V2@Q8I$213Q4
0350      V3@Q$I$7654B
0360 M12 *I6;SD6"Q6I$SD6"Q6"
0370      V2@Q4I$S18"H8
0380      V3@Q6$I$4Q1
0390 P53 R52
```

FIGURE #3



COMPONENT SIDE